Intelligent Agents & Decision-Making Homework 3

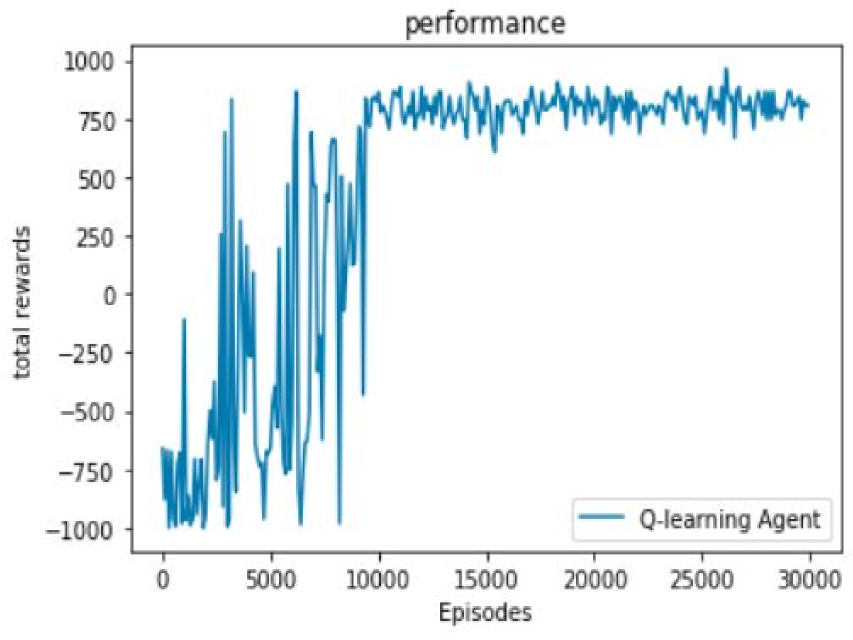
Aashish Adhikari, Parijat Bhatt

Question 1.

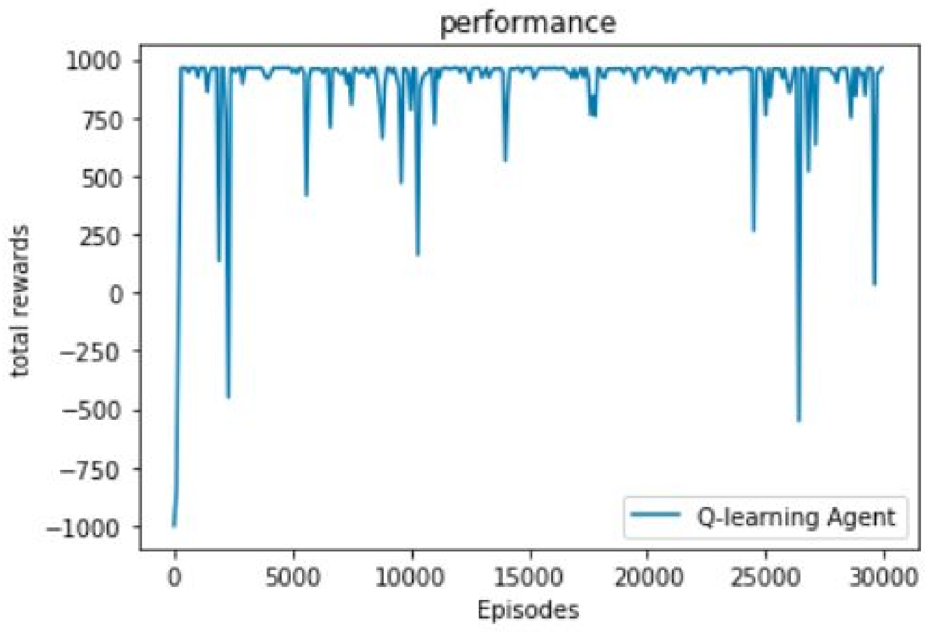
Answer:
Following are the graphs obtained for **Dangerous Hallway** and **16 * 16 Frozen Lake** Environments.
Note that we have included some figures for all combinations of parameters. However, **0.001** was the better learning rate in our case.

Q-Learning for Dangerous Hallway:

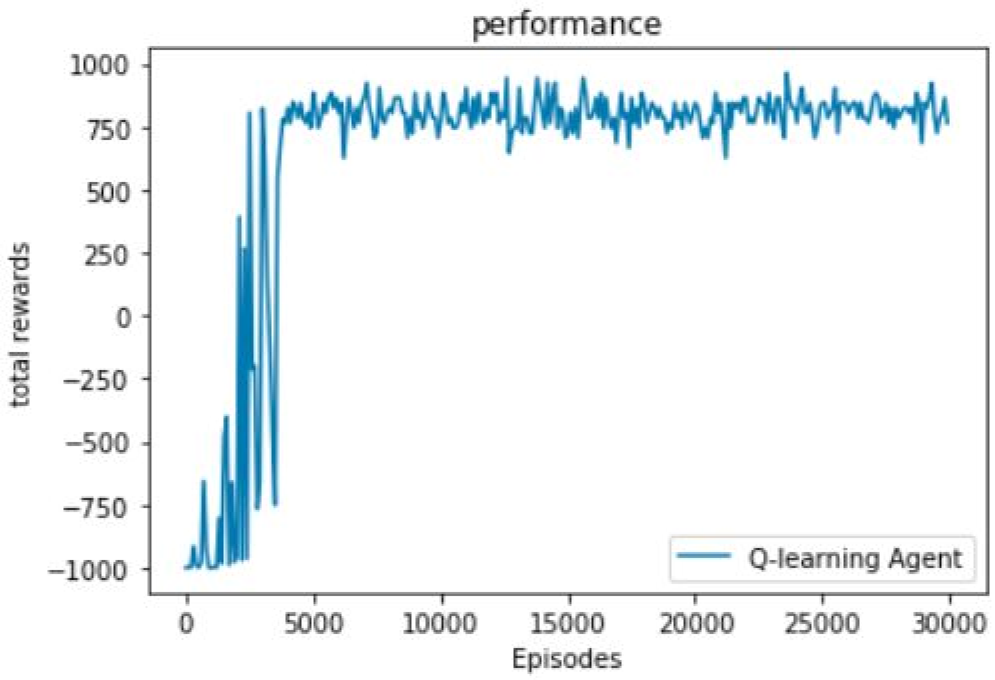
a).epsilon=0.3 and learning rate is 0.001



b)epsilon=0.05 and learning rate is 0.1

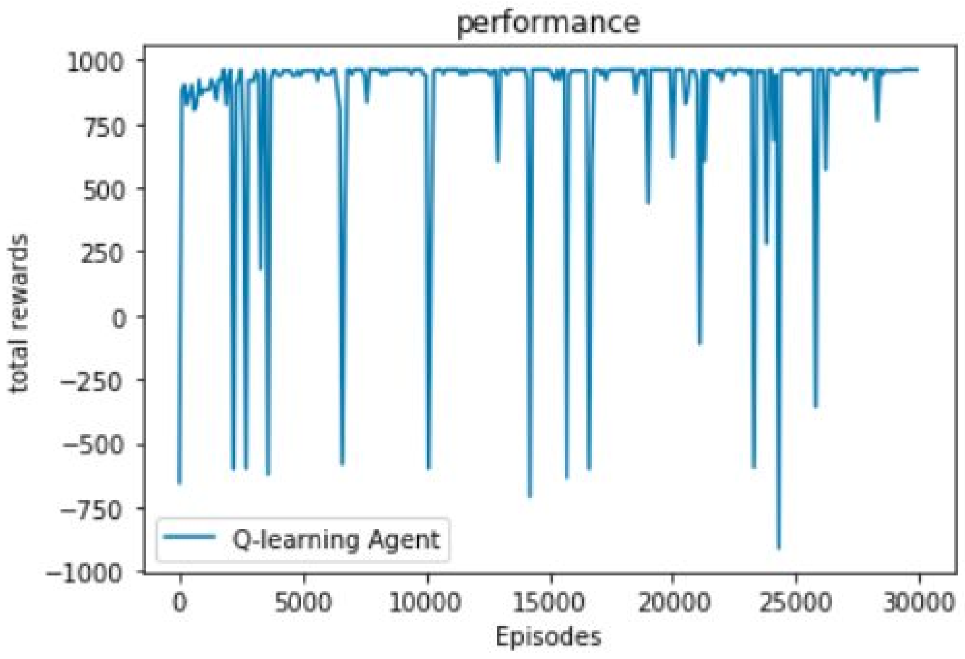


c) epsilon=0.3 and learning rate is 0.1



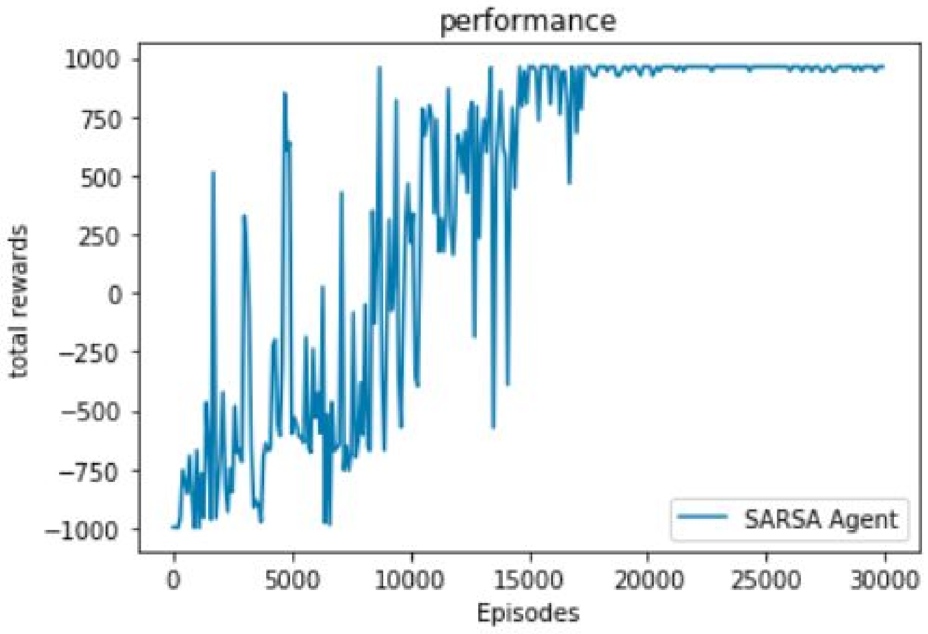
4.

d)<u>epsilon=0.05 and learning rate is 0.001</u>

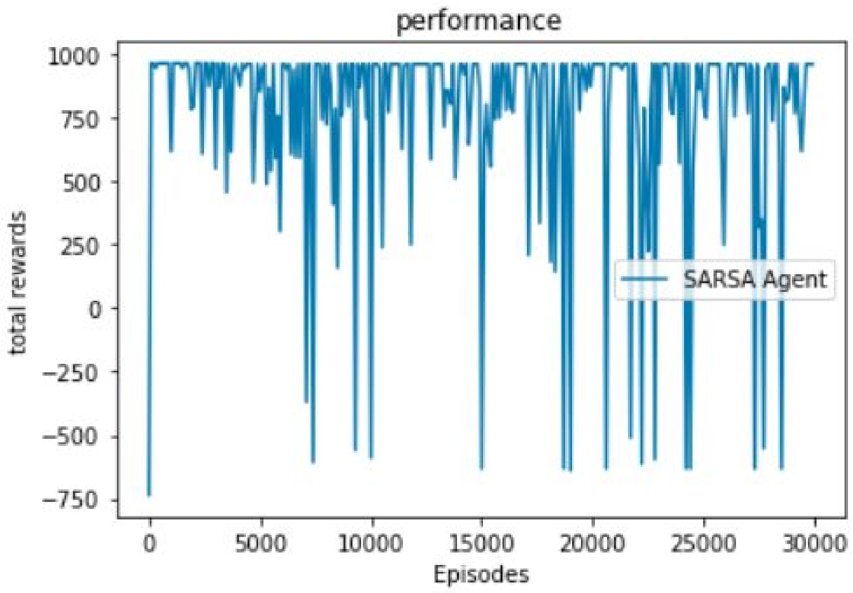


<u>SARSA for Dangerous Hallway:</u>

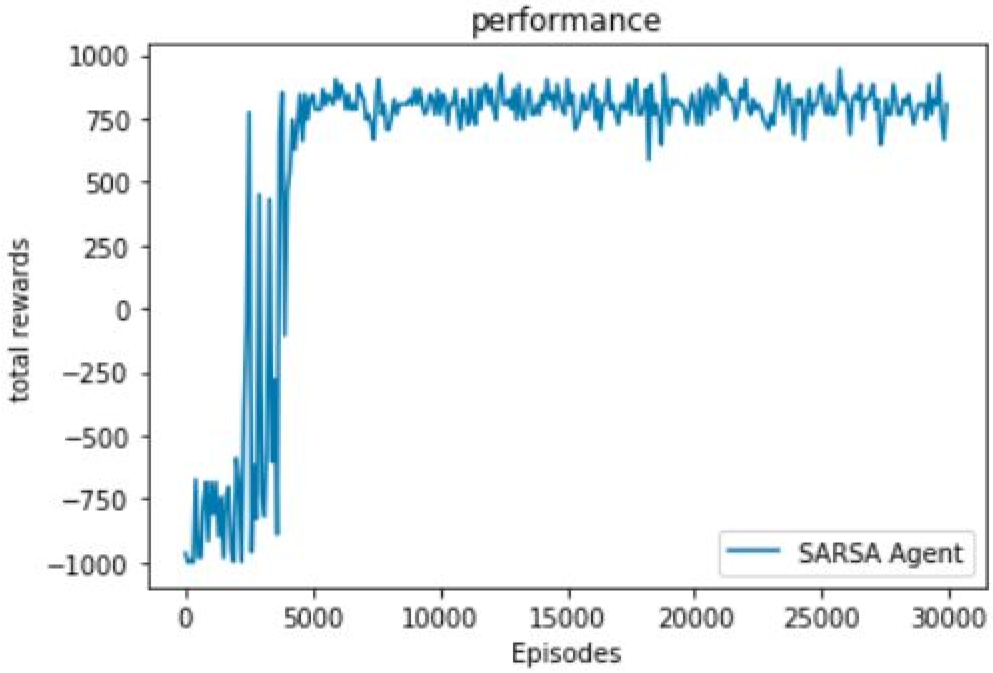
a<u>) epsilon=0.3 and learning rate is 0.001</u>



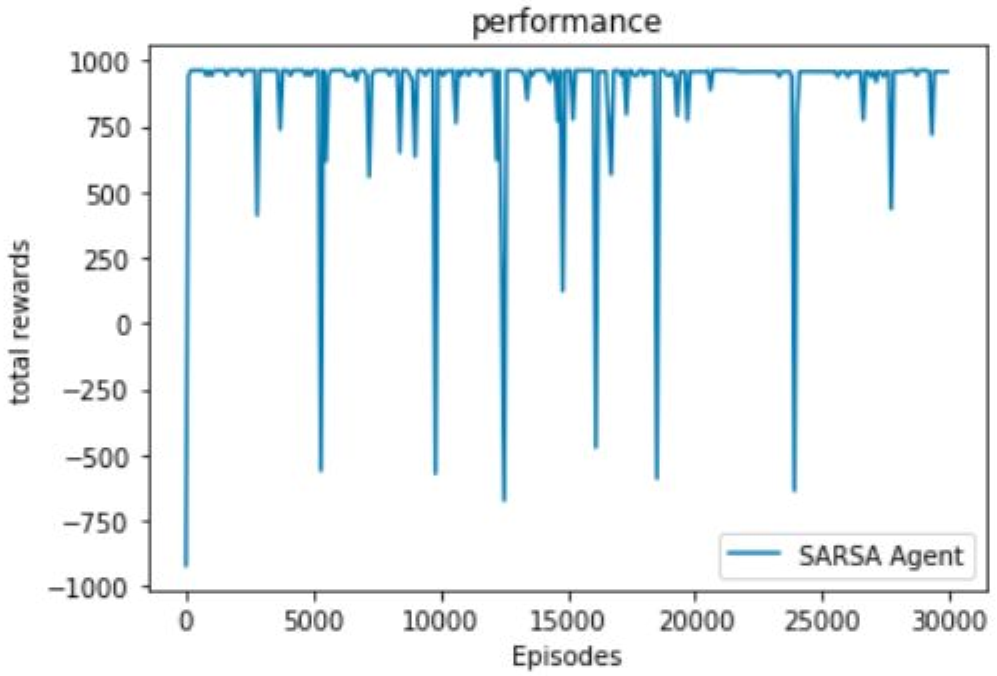
b) epsilon=0.05 and learning rate is 0.1



c. **epsilon=0.3 and learning rate is 0.1**

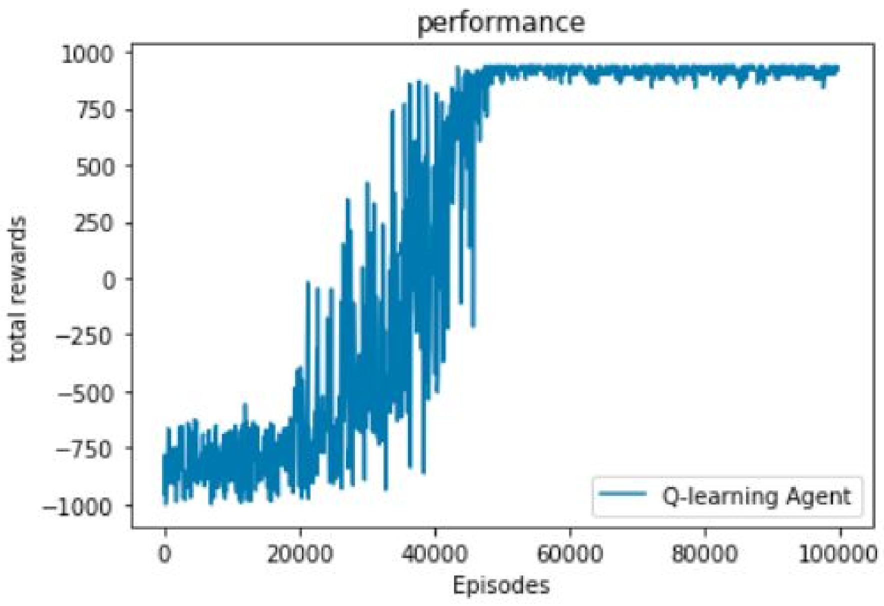


d)epsilon=0.05 and learning rate is 0.001

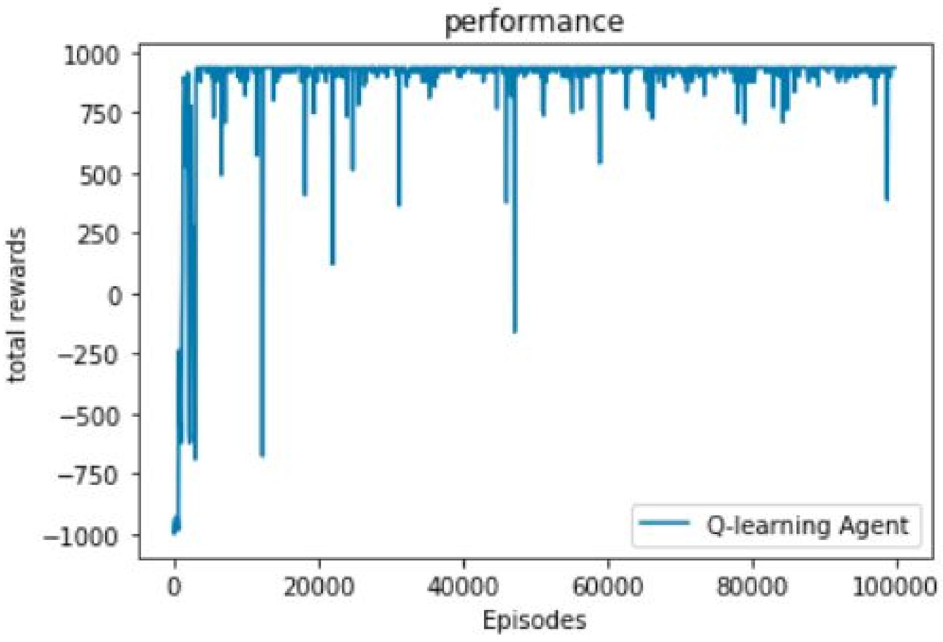

performance
1000
750
500
250
0
−250
−500
−750
−1000
total rewards
0
5000
10000
15000
20000
25000
30000
Episodes
SARSA Agent

Q-Learning for 16x16:

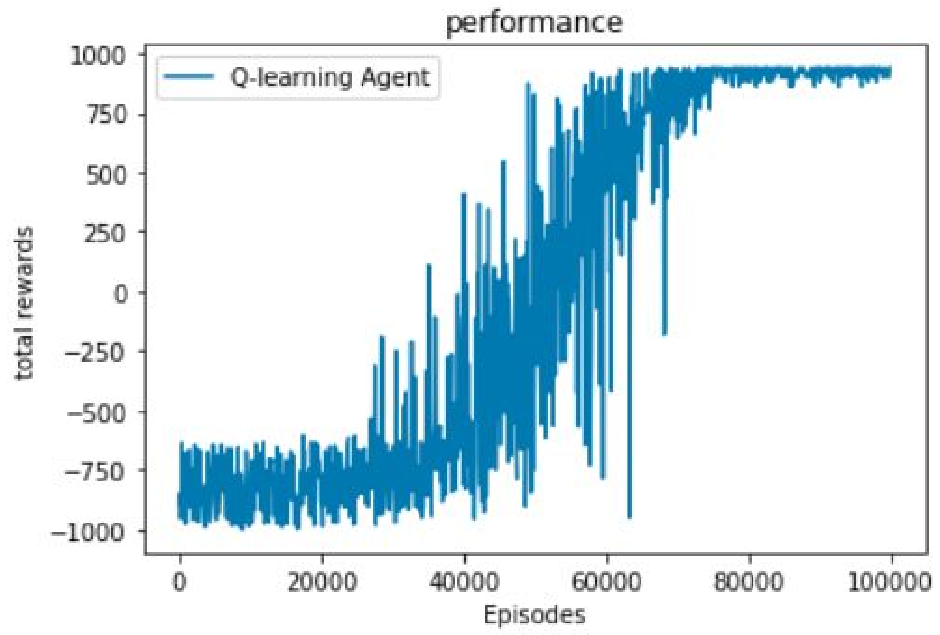
a) epsilon=0.3 and learning rate is 0.001



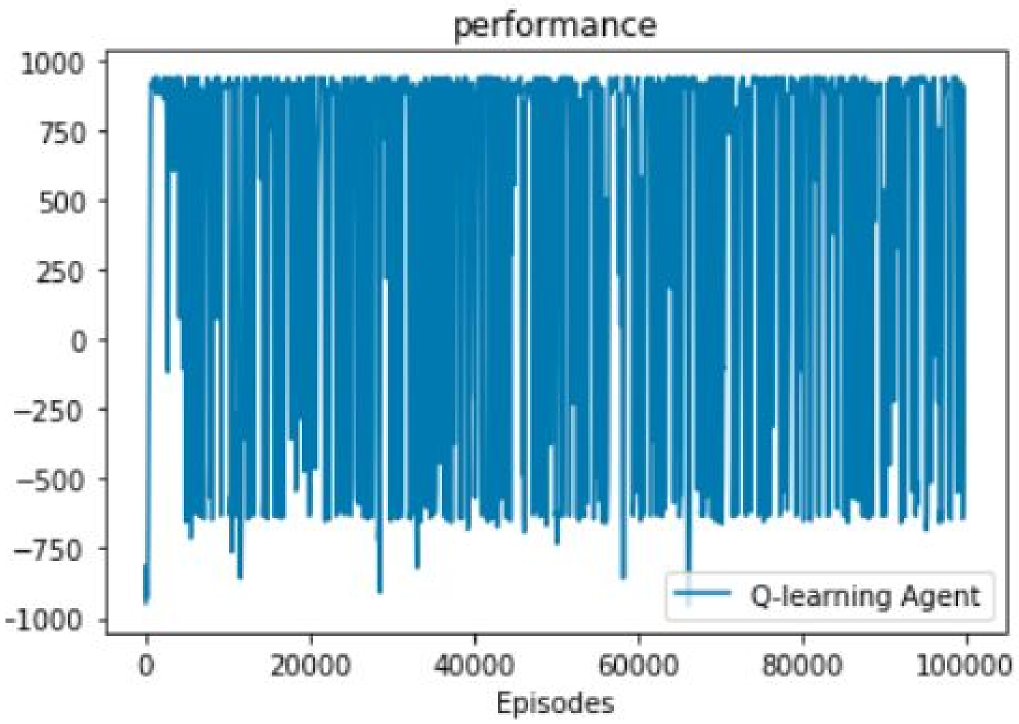
b) epsilon=0.3 and learning rate is 0.1



c) epsilon=0.05 and learning rate is 0.001

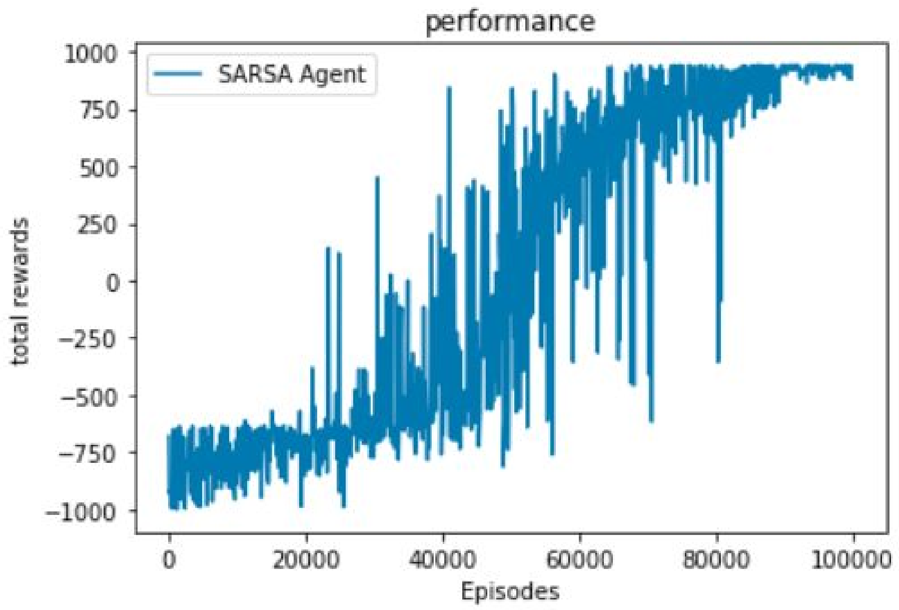


d) epsilon=0.05 and learning rate is 0.001

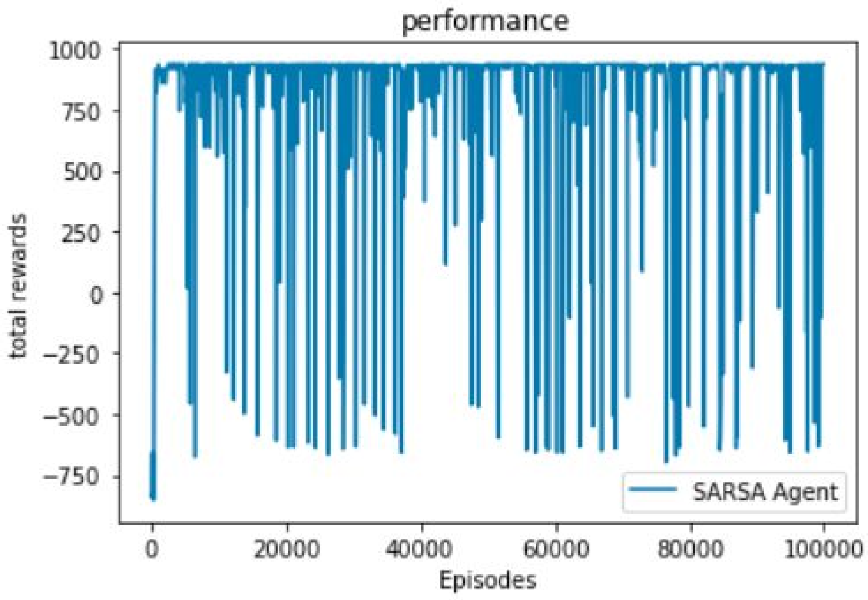


SARSA for 16x16:

a) epsilon=0.3 and learning rate is 0.001

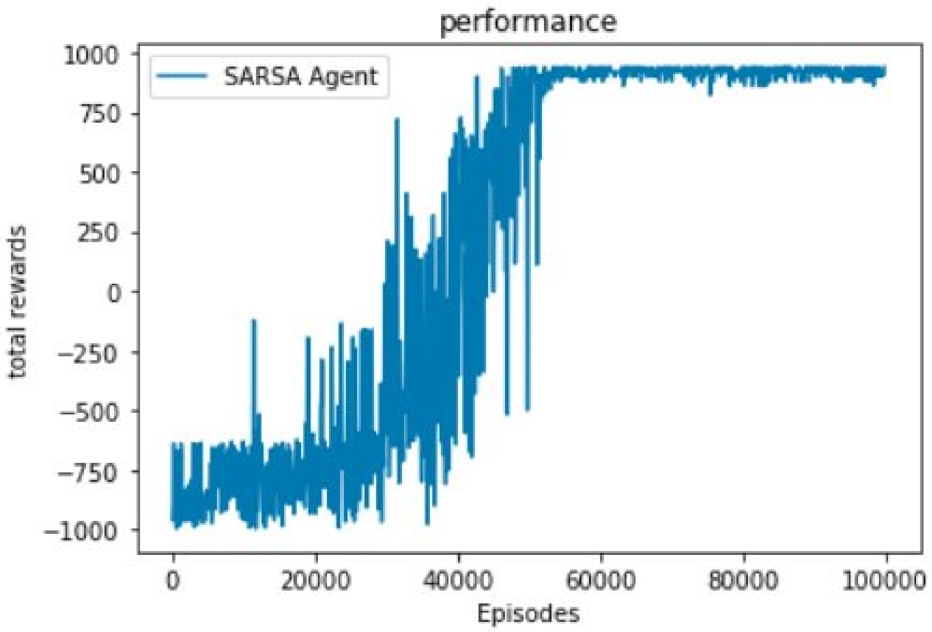


b) epsilon=0.05 and learning rate is 0.1

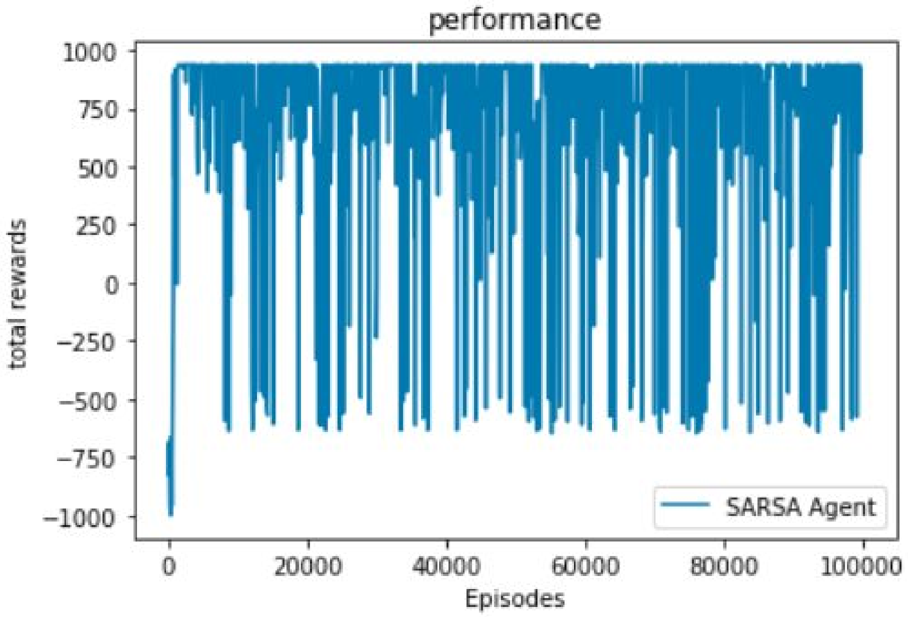


<u>c) epsilon=0.3 and learning rate is 0.001</u>

Figure size 432x288 with 0 Axes



<u>d) epsilon=0.05 and learning rate is 0.1</u>



Note: The explanations for the following questions are based on **Dangerous Hallway** unless mentioned otherwise.

Question 2.

While using the learning rate of 0.001, SARSA produced a very smooth learning curve eventually as seen in the figure. The learning time was 76 seconds. Also, in the grid, only the states in the proximity of the goal state displayed bright yellow color indicating high values. However, with the learning rate of 0.1, the learning curve was still very jittery till the end as compared to the learning rate 0.001. Also, the learning time was 53 seconds. Also, the distribution of yellow tiles was much prevalent.

The jittery nature is because for a high learning rate, the action value updates overestimate the update that need to be made. As learned in class, these algorithms need a sweet spot in terms of the learning rate for a good convergence. If the learning rate is not well-defined as in this case, the values overshoot and thus this can be seen in the learning curve behavior as well.The distribution of yellow tiles is much more because as mentioned above, overshooting results in over-estimation of the values that corresponds to higher Q values.

Question 3.

Q Learning produces interesting results with varying learning rates.
For LR = 0.001, it has a learning curve that eventually smoothes out as seen in the figure but is still jittery to some extent. However, for LR = 0.1, it appears to gain higher rewards fast. Also, for LR = 0.1, the learning curve, once settling to a plateau has a smooth nature with sharp spikes at different points because of the average reward going down as the overshooting Q values make sharp changes in the Q values at each update regardless of whether the update is positive or negative.

Question 4.

For epsilon = 0.3, the reward is very smooth after it has learned the policy. However, the reward is jittery for epsilon = 0.05. This is because smaller the epsilon, smaller the exploration. Epsilon is a form of adding noise in the system such that the agent takes non-greedy actions. If epsilon is near to zero, the agent does not take that many exploration steps and hence, the agent has less information about the environment since it gets a lesser chance of learning. Eventually, it makes less optimal choices. The key idea is that larger the exploration, more will the agent know about better policies.

Question 5.

For Q-learning, the reward is jittery for both epsilon values until the end. However, we can see that the agent has learned faster for epsilon 0.05. I.e., 19 seconds as compared to 43 for epsilon = 0.3. My assumption is that since Q learning's target is also greedy and lowering the epsilon value such that exploration also becomes more greedy coincide with each other and this leads to faster convergence. Also, the reason it is jittery is because Q learning uses greedy action for target estimation (while SARSA takes epsilon-greedy.) Hence, this target estimation is biased.

Question 6.

Yes. There is a difference in the policies learned.
Consider the learning rate of 0.001.
If you take epsilon 0.3, SARSA avoids the bridge and goes round the holes whereas Q Learning goes through the bridge.
On a side note, however, SARSA has learned to go through the bridge when we use the epsilon value of 0.05 instead of 0.3 because this reduction in exploration essentially makes it a greedy policy just like Q learning.

Q learning takes advantage of the greedy policy as it looks for the greedy actions to take for the next state. However, SARSA cannot take this advantage. Hence, the policies learned are different. Q LEARNING LEARNS TO GO THROUGH THE BRIDGE WHEREAS SARSA DOES NOT for epsilon = 0.3 and LR = 0.001.

Question 7.

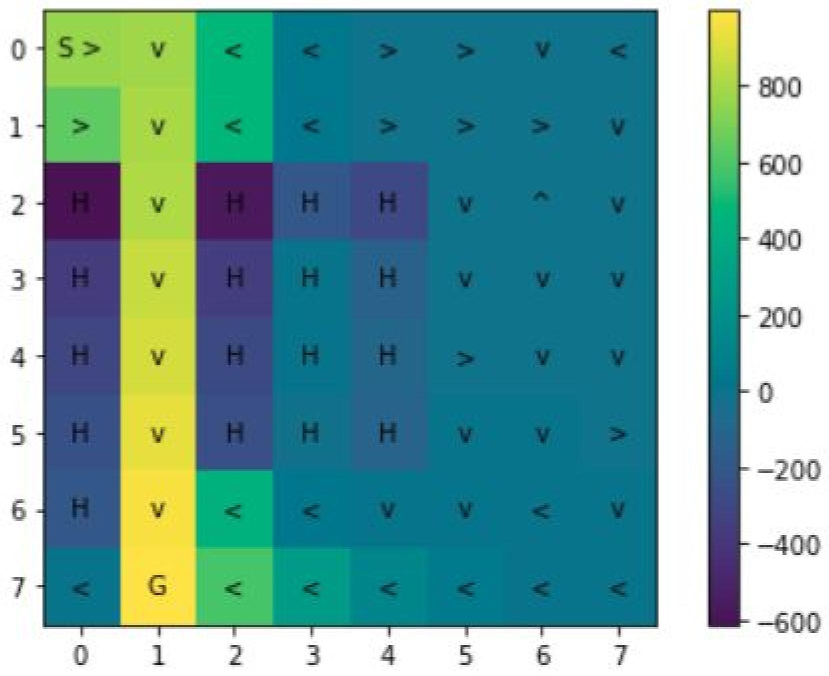
Note:The question mentions just one algorithm and we are using Q Learning in this case.

Shown below are the Q values and policy graphs for **Dangerous Hallway** map using distributed approach.

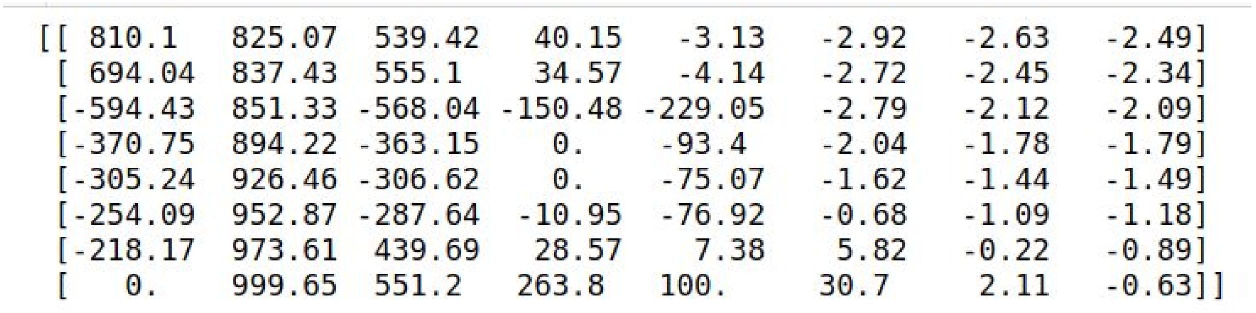
a) Dangerous Hallway

Distributed Approach

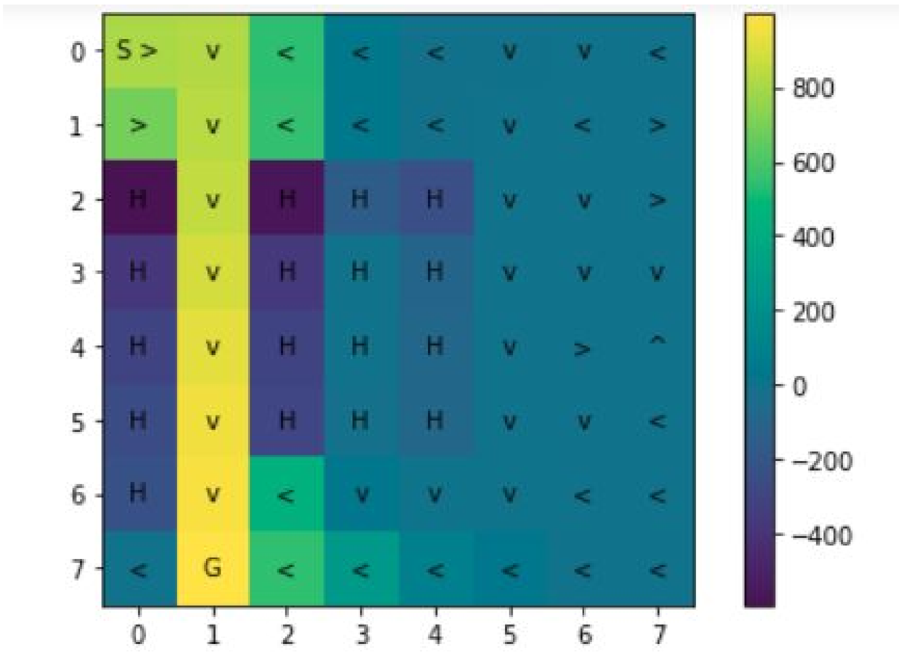
```
[[ 7.65e+02  7.85e+02  4.72e+02  2.65e+01 -4.25e+00 -3.73e+00 -3.39e+00 -3.21e+00]
 [ 6.46e+02  8.03e+02  4.72e+02  2.07e+01 -5.36e+00 -3.49e+00 -3.18e+00 -3.07e+00]
 [-6.12e+02  8.13e+02 -5.70e+02 -1.96e+02 -2.85e+02 -3.78e+00 -2.78e+00 -2.73e+00]
 [-3.64e+02  8.65e+02 -3.57e+02  0.00e+00 -1.22e+02 -2.78e+00 -2.40e+00 -2.40e+00]
 [-3.07e+02  8.95e+02 -2.74e+02  0.00e+00 -9.79e+01 -2.05e+00 -1.98e+00 -2.02e+00]
 [-2.31e+02  9.33e+02 -2.46e+02 -1.88e+01 -1.10e+02 -8.60e-01 -1.40e+00 -1.66e+00]
 [-1.98e+02  9.68e+02  4.34e+02  2.26e+01  8.05e+00  9.33e+00  2.58e-01 -1.39e+00]
 [ 0.00e+00  1.00e+03  5.90e+02  2.86e+02  1.22e+02  4.28e+01  4.37e+00 -7.59e-01]]
```



Non-distributed Approach



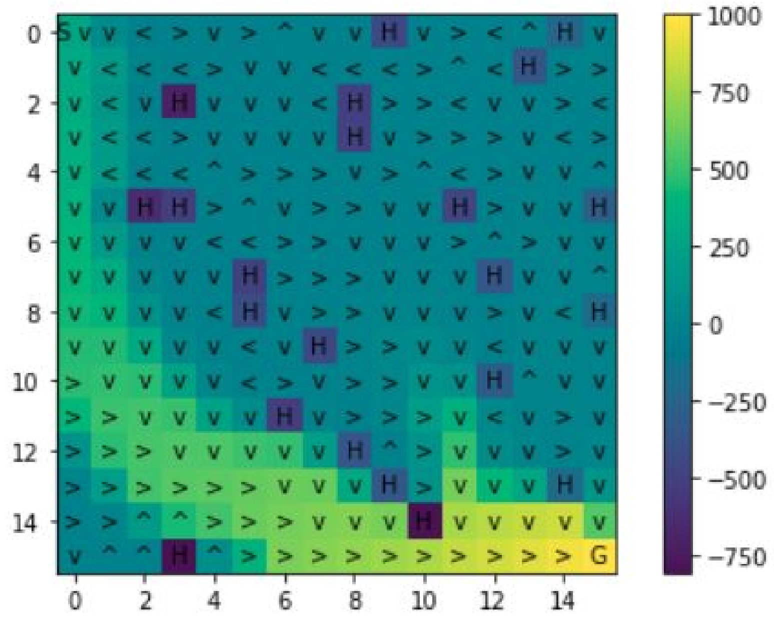
```
[[ 810.1    825.07  539.42   40.15   -3.13   -2.92   -2.63   -2.49]
 [ 694.04  837.43  555.1    34.57   -4.14   -2.72   -2.45   -2.34]
 [-594.43  851.33 -568.04 -150.48 -229.05   -2.79   -2.12   -2.09]
 [-370.75  894.22 -363.15    0.    -93.4    -2.04   -1.78   -1.79]
 [-305.24  926.46 -306.62    0.    -75.07   -1.62   -1.44   -1.49]
 [-254.09  952.87 -287.64  -10.95  -76.92   -0.68   -1.09   -1.18]
 [-218.17  973.61  439.69   28.57    7.38    5.82   -0.22   -0.89]
 [   0.    999.65  551.2   263.8   100.     30.7     2.11   -0.63]]
```



We can see that Q Learning produces similar values and policies Dangerous Hallway for both distributed and non-distributed approaches.
On a side note, although not illustrated here, SARSA produced somewhat different Q values and policies for distributed and non-distributed approaches.

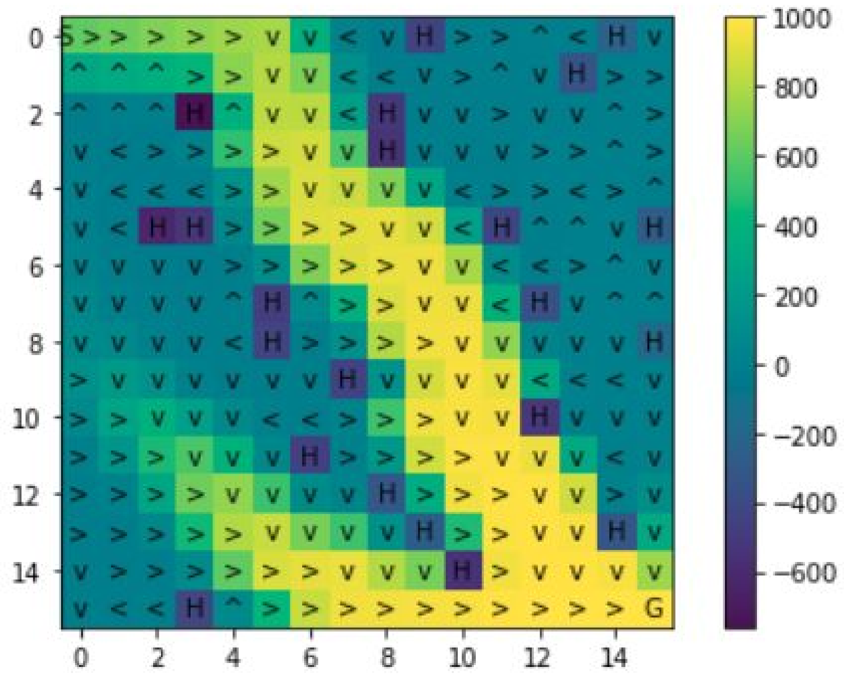
b) 16 by 16 map

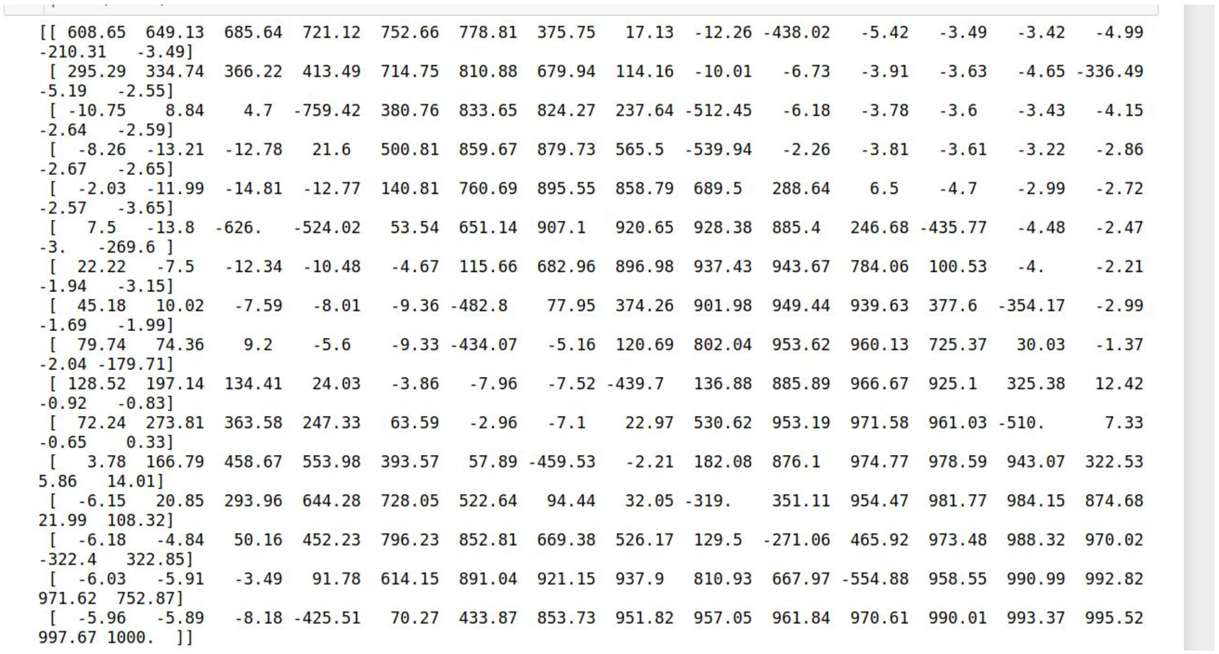
Distributed Approach



```
[[ 271.79   68.57  -11.65  -11.67  -10.82  -10.14   -9.55   -9.36  -11.46 -443.06   -5.29   -3.21   -3.11   -4.86
-232.9    -3.27]
 [ 296.61  181.75    6.73  -16.24  -10.26   -9.61   -9.25   -9.22  -10.3    -7.52   -3.57   -3.38   -4.46 -358.68
-4.98    -2.6 ]
 [ 318.69  156.02  -15.32 -715.39  -12.19   -8.95   -8.75  -10.95 -501.1    -6.27   -3.61   -3.44   -3.33   -4.72
-2.76    -2.67]
 [ 339.85  187.82   12.11  -14.52   -8.31   -8.33   -8.1   -10.04 -487.95   -5.64   -3.78   -3.47   -3.22   -2.93
-2.76    -2.7 ]
 [ 360.7   190.26   -9.31  -11.84   -8.11   -7.84   -7.44   -6.88   -7.33   -4.68   -4.22   -5.12   -3.15   -2.84
-2.74    -4.3 ]
 [ 379.99   67.1  -636.69 -513.91   -9.83   -7.41   -6.87   -6.32   -5.64   -4.99   -5.54 -466.51   -4.74   -2.57
-3.13 -297.55]
 [ 401.28  194.57   -7.08   -7.93   -6.68   -9.56   -6.37   -5.83   -5.03   -3.55   -4.02   -5.37   -3.98   -2.32
-2.17    -3.44]
 [ 421.58  309.19   21.85   -5.91   -8.53 -490.      -7.99   -5.1    -2.83    2.78   -0.47   -4.11 -354.17   -3.13
-1.88    -2.35]
 [ 440.73  400.49  121.57   -0.3    -7.61 -423.21   -6.74   -7.14   -2.57   13.32   15.86   -1.57   -3.48   -1.57
-2.36 -197.57]
 [ 458.59  459.31  311.65   45.73   -4.24   -6.47   -6.93 -434.64   -2.27   32.92   70.25   19.45   -2.83   -1.31
-1.12    -1.14]
 [ 474.81  491.09  475.1   249.5    23.72   -4.05   -5.62   -5.04   -2.01   22.48  127.47  115.55 -300.35   -1.31
-0.81    -0.35]
 [ 413.55  505.91  521.42  504.88  244.55   80.66 -506.56    2.91   -2.8    28.76  213.05  344.39   49.83    0.01
-0.34    4.43]
 [ 125.07  472.83  536.48  552.75  549.27  517.16  497.06  220.86 -352.88   -2.62  125.08  486.7    82.16   17.99
0.49   30.45]
 [   8.09  214.13  529.87  569.43  588.28  607.29  626.85  624.82  307.27 -340.46  120.63  647.48  422.85  285.26
-194.35  142.6 ]
 [  -3.87   12.23  197.59  445.08  528.1   620.03  649.36  671.08  688.2   659.92 -807.53  776.04  814.21  843.36
860.87  565.65]
 [  -3.95   -3.67   -7.59 -802.26   92.84  377.74  663.12  694.99  720.15  749.77  781.3   833.03  869.96  909.8
953.34 1000.  ]]
```

Non-distributed Approach





```
[[ 608.65  649.13  685.64  721.12  752.66  778.81  375.75   17.13  -12.26 -438.02   -5.42   -3.49   -3.42   -4.99
-210.31   -3.49]
 [ 295.29  334.74  366.22  413.49  714.75  810.88  679.94  114.16  -10.01   -6.73   -3.91   -3.63   -4.65 -336.49
-5.19   -2.55]
 [ -10.75    8.84    4.7  -759.42  380.76  833.65  824.27  237.64 -512.45   -6.18   -3.78   -3.6    -3.43   -4.15
-2.64   -2.59]
 [  -8.26  -13.21  -12.78   21.6   500.81  859.67  879.73  565.5  -539.94   -2.26   -3.81   -3.61   -3.22   -2.86
-2.67   -2.65]
 [  -2.03  -11.99  -14.81  -12.77  140.81  760.69  895.55  858.79  689.5   288.64    6.5    -4.7    -2.99   -2.72
-2.57   -3.65]
 [    7.5   -13.8  -626.   -524.02   53.54  651.14  907.1   920.65  928.38  885.4   246.68 -435.77   -4.48   -2.47
-3.    -269.6 ]
 [  22.22   -7.5   -12.34  -10.48   -4.67  115.66  682.96  896.98  937.43  943.67  784.06  100.53   -4.     -2.21
-1.94   -3.15]
 [  45.18   10.02   -7.59   -8.01   -9.36 -482.8    77.95  374.26  901.98  949.44  939.63  377.6  -354.17   -2.99
-1.69   -1.99]
 [  79.74   74.36    9.2    -5.6    -9.33 -434.07   -5.16  120.69  802.04  953.62  960.13  725.37   30.03   -1.37
-2.04 -179.71]
 [ 128.52  197.14  134.41   24.03   -3.86   -7.96   -7.52 -439.7   136.88  885.89  966.67  925.1   325.38   12.42
-0.92   -0.83]
 [  72.24  273.81  363.58  247.33   63.59   -2.96   -7.1    22.97  530.62  953.19  971.58  961.03 -510.      7.33
-0.65    0.33]
 [    3.78  166.79  458.67  553.98  393.57   57.89 -459.53   -2.21  182.08  876.1   974.77  978.59  943.07  322.53
5.86   14.01]
 [  -6.15   20.85  293.96  644.28  728.05  522.64   94.44   32.05 -319.    351.11  954.47  981.77  984.15  874.68
21.99  108.32]
 [  -6.18   -4.84   50.16  452.23  796.23  852.81  669.38  526.17  129.5  -271.06  465.92  973.48  988.32  970.02
-322.4   322.85]
 [  -6.03   -5.91   -3.49   91.78  614.15  891.04  921.15  937.9   810.93  667.97 -554.88  958.55  990.99  992.82
971.62  752.87]
 [  -5.96   -5.89   -8.18 -425.51   70.27  433.87  853.73  951.82  957.05  961.84  970.61  990.01  993.37  995.52
997.67 1000.  ]]
```

It is observed that the distributed approach has very different policy and Q values when compared to non-distributed Q Learning. Also, while running the experiment for SARSA,

it produced different Q values and policies for distributed and non-distributed approaches.

Question 8.

Following table compares the time taken for the algorithm to run for "**Dangerous Hallway**" environment with evaluation turned on.

	Single-Core	Distributed (2,4)	Distributed(4,4)	Distributed(8,4)
Dangerous Hall	57.00	8.9470	5.5060	8.5252
Frozen Lake (16 * 16)	536.2269	Server Disconnection each time*	Server Disconnection each time*	Server Disconnection each time*

*****For some reason, the server disconnects after running for a while for the configurations mentioned. I tried running it for less iterations (10000, 20000, 40000, 50000). It works for upto 20000 but then the server disconnects after some while for more episodes. Hence, I will explain on the basis of Dangerous Hall.

The performance of (2,4) distributed configuration is significantly faster than the single core performance. The performance increases with the increase of the collector workers to 4. However, as we increase the number of collectors to 8, the performance decreases. It is because the communication overhead between 8 collectors overwhelms the performance increase gained.

However, I also believe that the server availability/ users accessing the server affects this and we could see some different trend in such cases.

Question 9.

Following table compares the time taken for the algorithm to run for "Dangerous Hallway" and "Frozen Lake 16 * 16" environments with evaluation turned off.

	Single-Core	Distributed (2,4)	Distributed(4,4)	Distributed(8,4)
Dangerous Hall	47.87	8.539	6.2081	5.0311
Frozen Lake (16 * 16)	429.232	138.849	122.339	Server Disconnection

				each time*

The general trend here is that with the increase in the number of collection workers, the time taken goes down. It is to note, however, that these readings depend upon the server usage and thus might not be consistent. For instance, I recall getting a (2,4) run to be 105.524 seconds which is faster than (4,4) run mentioned above.

*****Instead of (8,4), I tried (6,4) and it returned a learning time of 46.1032.
Nonetheless, I ran this experiment multiple times and noticed that the time taken has quite some variance. I am assuming this depends on the server usage by multiple users.

In comparison to question 8, there is lesser time at each configuration given the fact that we are not running the evaluation code as mentioned. This significantly reduces the effective time complexity as we are doing less computation overall since we are effectively by-passing all the interleaved evaluation code.